GNC 2008
7th International ESA Conference on Guidance, Navigation & Control Systems
2-5 June 2008, Tralee, County Kerry, Ireland

THE FEASIBILITY OF APPLYING PLUG-AND-PLAY CONCEPTS TO SPACECRAFT GUIDANCE, NAVIGATION, AND CONTROL SYSTEMS TO MEET THE CHALLENGES OF FUTURE RESPONSIVE MISSIONS

L. J. Hansen[1], P. Graven[2], D. Fogle[3], J. Lyke[3]
[1]HRP Systems, USA. [2]Microcosm, Inc., USA. [3]Air Force Research Laboratory (AFRL), USA
hansen@hrpsystems.com

ABSTRACT

In this paper, a methodology for dramatically reducing the time required to build a complex system based on the self-organization of self-describing components is presented. In this "plug-and-play" approach, component services are exposed for use through mechanisms similar to those in a "publish-subscribe" framework. The associated framework provides convenient mechanisms for organizing application domains (such as guidance, navigation, and control) along a hierarchy of components. The hierarchy of components, which are themselves self-descriptive, is non-unique but enforced by convention and discipline. This paper describes the motivations for this work and elements of its current implementation as a disruptive technology.

INTRODUCTION

Spacecraft development has become expensive and protracted. Expense begets expense, one of many manifestations being intolerance for failure, further driving higher costs/time and more intolerance for failure. A movement within the Department of Defense, referred to as "Operationally Responsive Space" (ORS) is attempting to promote a conceptual shift to spacecraft that can be built much faster with "good enough" (versus optimal) performance, emphasizing the pace of creation as having prime importance. One approach to achieve this speed involves a discipline around pre-built, modular components, capable of flexible [re]configuration through (mostly) software to meet a diverse range of mission requirements. Modular standards are not sufficient, however. Recently, the use of plug-and-play (PnP) mechanisms, similar to that in the personal computer, have been studied for ORS [1-2], and an entire spacecraft is being developed based on these principles [3]. PnP approaches hide complexity through machine-negotiated integration. They constitute an open systems architecture, which can expand the notion of standardization to level deeper than the aerospace industry has ever attempted. This open systems architecture allows a common interface abstraction in which multiple vendors can create modular, even proprietary, components that will interoperate. This simple idea is revolutionary in that complexity can be reduced to the ability to generate, standardize, and operate through interface abstractions. This vision of implementing PnP in any system (not only spacecraft) requires not simply new thinking, but judicious engineering as complexities associated with the various components will be hidden, but they will still exist.

PnP approaches are not without challenges. The benefits from PnP come through the implementation of additional circuitry and software, which is a form of overhead. The decoupling implied by "modularity" may result in compromises compared to tightly-coupled designs (e.g. timing, latency, and throughput). Rapidly-assembled systems may lack precision alignment. It is hoped, however, that these early implementations will be "good enough" pathfinders to move the industry forward and evolve the technology to eventually approach, even exceed the performance of existing systems. In this way, PnP, as applied to the spacecraft development in general and guidance, navigation, and control (GNC) components in particular, represents a disruptive technology[1].

The concepts involving the form of PnP discussed in this paper are broad, spanning hardware, software, tools, and systems engineering approaches that are expressed through them. The Air Force Research Laboratory (AFRL) has cultivated a set of these ideas under the banner of "Space PnP Avionics" (SPA). The domains of application can be a broad as a network of platforms, to include its operation and user segments, or as narrow as a small array of scalar point sensors. Microcosm Inc., in conjunction with HRP Systems, has worked with AFRL under Small Business Innovative Research (SBIR) contracts to develop prototype implementations focused on guidance, navigation, and control (GNC) as a narrowly-focused PnP application domain. The early objective of the work examined a self-configuring network, with features including resource discovery, component configuration and management, and system level configuration and graceful degradation (fault tolerance) based on the dynamic reconfiguration of the system GNC applications to dynamically accommodate available PnP components.

---

[1] The term disruptive technology was coined by Clayton M. Christensen (Harvard Business School Professor) [4] Essentially, disruptive technologies are "innovations that result in worse product performance except for niche applications in the near-term, but ultimately provide tremendous utility in the broader market."

This paper is organized as follows. The concept of "operationally responsive space" as providing motivation for the present work is provided in summary. Next, the framework of PnP and application of PnP concepts to spacecraft avionics system development are discussed. Finally, the viability of applying PnP concepts to GNC system is appraised, with specific examples taken from several of Microcosm's implementations of PnP GNC hardware and software demonstrations.

OPERATIONALLY RESPONSIVE SPACE

It is not uncommon for spacecraft to cost billions of dollars, and in most cases the cost overruns for such systems can also cost millions of dollars and years of development timeline. In each case, these spacecraft are tailored or "optimized" to meet specific mission objectives and the derived requirements. Often the pursuit for "optimal" performance eschews flexibility due to the overhead that comes with it. Unfortunately, lack of flexibility becomes a tremendous liability in a protracted development schedule, resulting in additional cost and time to deal with requirement changes. The problem has become significant enough to draw widespread attention with the aerospace community, and "operationally responsive space" (ORS) was coined in part as a counter-movement to protracted spacecraft developments. ORS has become the subject of entire conferences2 and the name of an office recently established by the Department of Defense [6]. ORS has meant "many things to many people", with the one invariant being speed in translating a user's need for a mission into a fielded system implementing that mission. Three responsiveness "tiers", referring to timescales, have been defined [7]. Tier 1 refers to responsiveness scales measured in minutes to hours. Tier 2 is identified as approaches that can be mobilized in "days-to-weeks". Tier 3 solutions are those that require longer times. This ORS-defined vision, mercifully bereft of implementation details, has been left free for researchers and planners to interpret. It is, for example, generally expected that a Tier 1 solution is based on some approach for repurposing existing fielded systems. Tier 3 systems, at the other extreme, are consistent with typical research timescales for creating proof-of-principle prototypes based on ideas with a reasonably solid technical basis. The sentiments of these concepts are notionally depicted in Fig. 1. It is "Tier 2" that provides the most interesting new regime for creative development, a timescale too aggressive (in a conventional sense) for building components for scratch, but a timescale compatible with an incredibly efficient integration of pre-built components to form a deployable system.

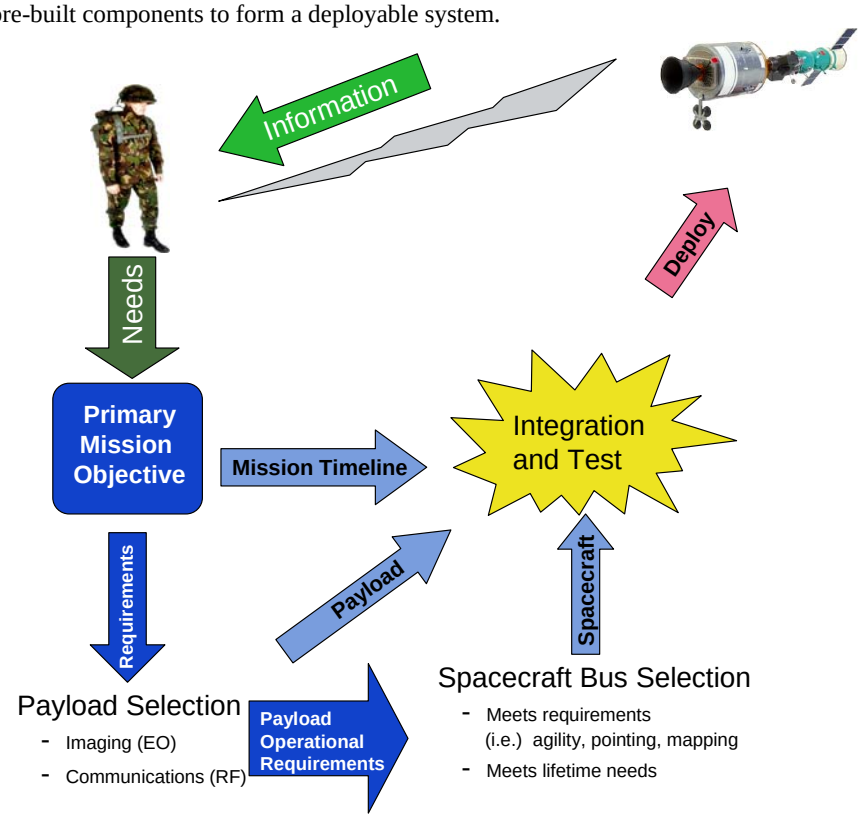

Fig. 1. Meeting the Needs of a War-Fighter through Operationally Responsive Space

[2] The Responsive Space conference, sponsored by AIAA LA and OC Sections, has been held in the Los Angeles area annually since 2003 (http://www.responsivespace.com).

To achieve ORS, elements of commercially accepted philosophies such as standardization, open architecture, commonality, and reuse are now being investigated by spacecraft manufactures and their customers. There are multiple ways to implement these philosophies in the evolving spacecraft development paradigm, such as the use of well defined spacecraft bus configuration to meet the needs of categorized mission requirements, developing an inventory of standard spacecraft bus structures that are ready to be mated with a mission specific payload, creating a modular set of sub-assemblies that are used to build a spacecraft bus, and implementing plug-and-play (PnP) at the component or subsystem level for rapid integration.

Mission requirements for ORS will typically fall into two categories, imaging and communications (other possibilities exist, such as space weather). For each of these mission types, more specific spacecraft bus requirements can be derived, such as data storage requirements, downlink requirements, pointing/mapping accuracy requirements, on-orbit life time, required margins, and launch environment constraints. In turn, each of these top level derived requirements will flow down into subsystem requirements. Eventually, the requirements can be grouped and a finite set of mission categories, with specific spacecraft bus requirements, can be established. These studies have been, and are being conducted as part of the ORS Program. To meet the intent of ORS, spacecraft bus manufacturers can develop an inventory of spacecraft bus assemblies that meet each of these mission categories, or they can acquire the common components between the spacecraft busses associated with the mission categories and be prepared to assemble a vehicle in 6 – 10 days. A spacecraft and payload, developed for a specific mission type and placed in inventory for future launch, might achieve rapid response. A standardized bus might be considered a modular component as it is paired with an "inventory" payload, which together are capable of meeting a mission objective when integrated and launched. Modularity of the spacecraft bus in terms of subsystems that are selected and combined to meet specific mission needs is another approach to creating off-the-shelf pieces that can be assembled to create a spacecraft.

SPACE PLUG-AND-PLAY AVIONICS (SPA)

The Tier 2 definition from the ORS vision has become the rallying theme of the research called "space plug-and-play avionics" (SPA) [2]. SPA, as a research program, attempts to tackle complexity through intelligent component and software strategies very similar to those used in commercial systems, but focused on embedded, fault-tolerant platforms. SPA was created primarily in response to the integration barriers caused by disparities in component interfaces. Spacecraft, as a class of complex system, necessarily involves the eventual convergence of many hardware and software components developed by different groups at different places and times. In space systems, an often exaggerated emphasis is placed on the use of "legacy" components, which are believed "safe" choices because they have been shown to work successful in some previously flown system. Legacy components, though an attractive proposition, do not always involve the same electrical interfaces. Components with such hardware differences, such as the case when a component with a MIL-STD-1553 interface needs to be connected to another component with a RS-422 interface, represent the most obvious form of disconnect, which is resolved by either reprocuring a component with an altered interface (which "destroys" the legacy benefit to some degree) or engineering a glue of hardware and software to bridge the dissimilar components. Software differences seem more subtle, such as the case when two RS-422 components are connected, but cannot work together due to differences in the ad hoc protocols of command and data handling. Once again, the disconnects are resolved through custom engineering, which seems tenable since "it is only software". On the scale of two components, resolving such disparities can be quite involved, but on the scale of an entire spacecraft, in which many such components must be unified into a coherent whole, it can be more than challenging. It is at one level no wonder that a large spacecraft can experience significant overruns.

In SPA, the attempt is to tackle the hidden layers of complexity underlying complex systems through a set of coping strategies, ranging from standard electrical interfaces and software protocols to self-organization mechanisms that dramatically reduce the need for custom engineering bridges between disparate components. Random arrangements of RS-422 components will likely never work without extensive engineering, but a collection of SPA components could be swiftly aggregated into a network that would self-organize in a way that can be clearly accessed through software of a compatible design (i.e., SPA-aware). Components "serve" their own descriptions to systems, using electronic datasheets. Many components have single connections, a SPA interface containing power, data, command, and timing connections. The overarching attempt is to create a constrained universe with SPA in which hardware and software elements can be freely composed to form systems of nearly arbitrary sophistication with minimal burdens on the humans who must build those systems.

SPA can be divided into two sets of technologies, the first being SPA interfaces and the second being SPA software. SPA interfaces attempt to provide a simple modularity approach for spacecraft components, as depicted in Fig. 2, which suggests a correlation between the plug-and-play models used in personal computer and "SPA-enabled"

spacecraft. In the case of the personal computer (PC), a host "platform" can support the addition of a number of ordinary components, such as mice, keyboards, and thumbdrives, using a single (USB) interface. The single interface supports power distribution, command, and data handling. Components in the PC "plug-and-play" through a process that supports the automatic enumeration of components and a brokering mechanism that seeks to match pre-written software drivers to devices. Within the component, an interface chip exists that on one "side" provides a compliant implementation of the USB standard3 and on the other "side" provides a convenient breakout interface. Individual device developers mate their custom componentry to this breakout interface and physically wrapper the device, resulting in a very simple presentation of a modular component to a user. The user is largely unaware and indifferent to the processes that make it possible to unite these PnP devices to the PC. In the SPA analogy (right half of Fig. 2), the spacecraft (as a platform) supports the addition of modular components, which are automatically recognized when connected to the spacecraft. The recognition is possible by encapsulating the component with an appliqué sensor interface module (ASIM), which plays a role similar to the USB interface "chip" in a mouse or keyboard. The ASIM contains an internal microcontroller, state machines, and memory and is intended to provide a convenient breakout interface to the "raw" custom circuitry of typical spacecraft components. When "wrappered" with an ASIM, a spacecraft component is referred to as a SPA component. All SPA components are self-describing through an electronic datasheet contained in the ASIM. The electronic datasheets in SPA are called extended transducer electronic datasheets (xTEDS), inspired by (but different from) the TEDS associated with the IEEE 1451 series of smart sensor standards [8]. The xTEDS is a powerful abstraction for a driverless form of plug-and-play, intended to provide some insularity from operating systems, since there are not dominant standards for aerospace. The xTEDS can be thought of as a "service description" for components, and in fact, functions not detailed in the xTEDS are effectively invisible to SPA.

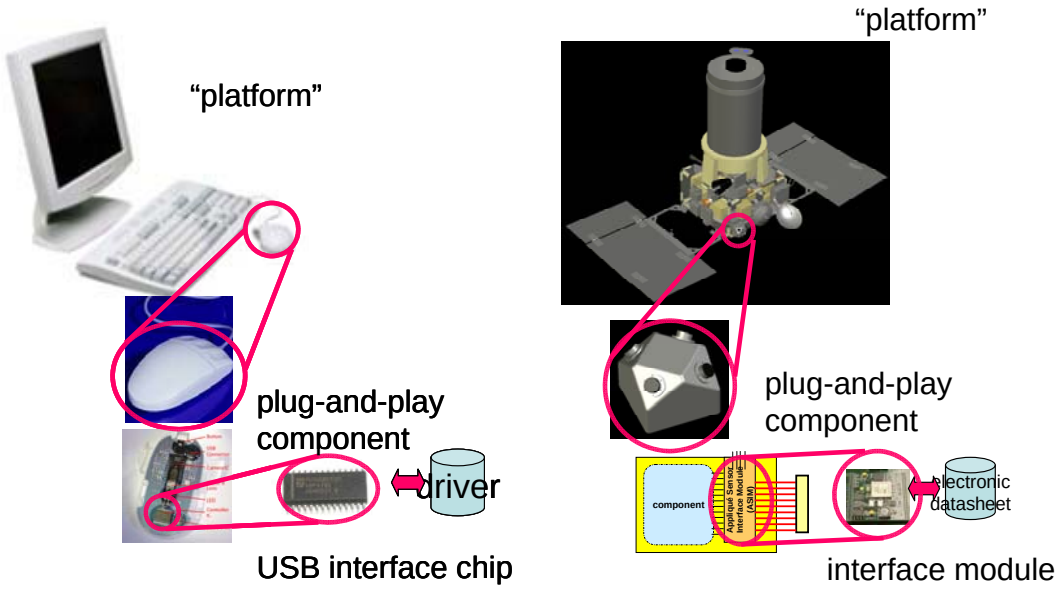

Fig. 2. Analog between personal computer and spacecraft based on plug-and-play principles

A number of physical interfaces have been studied for use in SPA, including USB [1] and Spacewire [9], which are denoted SPA-U and SPA-S, respectively. They differ from the standards they are derived from only in that it is necessary to provide supplements for power handling (i.e. up to 4.5A at 28VDC) and synchronization (1 PPS through two RS-422 conductors). Also, while USB provides for automatic enumeration of devices added to a USB network, Spacewire does not intrinsically support automatic organization of networks, and a protocol has been added to supplement PnP functionality. In principle, any number of SPA networks might co-exist in a complex system, mediated through bridge interfaces as necessary.

While SPA interface simplify the mechanisms for defining and networking PnP devices, a specialized software infrastructure is at the heart of the SPA concept. The fundamental software system is called the Satellite Data Model (SDM) [10]. It is sometimes called a middleware, but this distinction has been argued, since it is not always necessary for transactions between SPA components to "go through" SDM. SDM is a set of software components that provide a unified plug-and-play mechanism for "discovery and join", which involves registering components

[3] The USB standards are maintained by the USB Implementer's Forum (http://www.usb.org).

and services and implementing a "look-up service", which allows the producers and providers of these services to be located by software components that need them through schema queries (improved regular expression support has been recently added). SDM is comprised of four modules. The Data Manager implements the "registry" and look-up service functionality. Other SDM modules include the task manager, network manager, and processor manager. The SDM architecture is intended for distributed implementation, meaning that components can be dispersed throughout a network of processors within a SPA system. The data manager is the most important module, and while multiple copies may exist within a network, a single copy is active at any instant (other copies can be rapidly and automatically activated with an instrinsic dynamic mastering protocol).

SPA, with its SDM infrastructure, promotes software reuse at the application level. The new architecture is data-centric and relies on the abstraction of the core algorithms from the specific sensor and actuator suites that are selected for each mission. Ideally, applications can be prewritten, to include electronic datasheets (in this respect, SDM does not differentiate the origin of services as being distinctly hardware or software), to have PnP "awareness". Innovation in development is required to support this PnP "awareness", ranging from process-oriented enhancements to actual software implementations that facilitate new missions [11]. The resulting implementation requires that partitioning of software and data be at the lowest levels, which has given rise to several new concepts, such as atomic data elements and helper applications software modules (helper apps) [12] Additionally, the core algorithms must be designed to be generic and the software must be developed and tested, before the specifics of the mission may have been identified. Mission parameters associated with the vehicle configuration and characteristics must be easily tailored without requiring additional testing to facilitate the reuse of these generic core algorithms.

In the pursuit of engineering applications to "be" PnP, new software structures and concepts, such as subsystem controllers and activity agents have been introduced. Subsystem controllers are software applications that manage the subsystem activities. Activity Agents are software applications that exploit subsystems to perform specific tasks. The AFRL PnP concepts, illustrated in the software architecture shown in Fig. 3, have been prototyped and evaluated as part of the on-going activities in the responsive space testbed (RST), a dedicated laboratory at Kirtland Air Force Base for responsive space research. The software architecture, along with currently available hardware component and payload devices that could be available as off-the-shelf, can be configured to create a spacecraft that nominally meets the requirements of the mission.

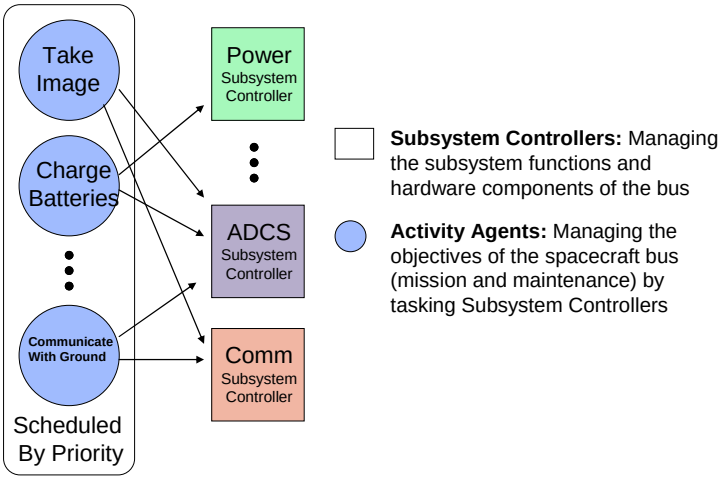

Fig. 3. An architecture for applications in SPA/SDM

PLUG-AND-PLAY (PNP) GUIDANCE, NAVIGATION, AND CONTROL (GNC)

While PnP approaches are expected to have dramatic benefits to space systems, one concern that constrains PnP is that many spacecraft subsystems, guidance, navigation, and control (GNC) in particular, have stringent real-time requirements that may not facilitate the use of PnP concepts. The intrinsic determinism PnP may not achieve the rigorous standards of traditional spacecraft in terms of timing, latency, and throughput, but may be "good enough" [14]. In this section, PnP, as focused in the domain of GNC, is presented.

As PnP Avionics become more available, PnP architectures are accepted and PnP middleware software, used to aid in data discover and transmission, is developed and validated, the application of PnP to guidance, navigation, and control (GNC) needs to be developed and validated. Microcosm, and team member HRP Systems, have been

developing approaches and evaluating the viability of PnP applied to GNC systems for over 8 years, under the direction of AFRL technical personnel. GNC provides a particularly challenging test case for modularity and PnP for a number of reasons:

- Mission criticality
- Real-time operation / Low tolerance for latency
- Very high availability and reliability requirements
- Increasing bandwidth and computational requirements
- Complexity
- Coordination requirements

To reach balance between the desire for ORS and the traditional criteria for developing a GNC system, the Microcosm team established the following high level requirements for the implementation of a PnP architecture for GNC applications:

1. Real-time (predictive, if not deterministic) performance
2. Support high availability, capable of fault tolerance
3. Scalable and extensible design (to both varying levels of network bandwidth and higher capability processors)
4. Low hardware overhead — size, weight, power
5. Compact software (to maximize the use of low cost microcontrollers)
6. Leverage existing technology (hardware, software, protocols)
7. Simplify system design and integration process
8. Maximize reuse [through well defined services and application program interfaces (APIs)]
9. Maximize portability (through partitioning of platform dependent code)
10. Create a freely available, no royalty, open design specification

Based on these requirements developed in the 2004 – 2005 time period, the Microcosm team established a demonstration of a self-configuring network for GNC applications. The demonstration hardware is shown in Fig. 4. In this prototype environment, the components of a system are rapidly assembled with minimal need to write detailed, low-level code pertaining to the interface or usage of each element. Additionally, with standardized data interfaces, independent of the source and the algorithms, control law software and algorithms are quickly assembled and integrated for varying sensor and actuator suites. The salient features of this demonstration were the use of multi-level networks, tailored for high and low speed (as well as in between) data transmission rates, the incarnation of a standard data protocol that allows data elements to be efficiently announced to the system, the use of a central data manager to broker the data from publishers to subscribers, the use of point-to-point communications when the data brokering was complete, and abstraction of the core control algorithms from the specific sensor and actuator suite. These important elements of the initial demonstration have been enhanced, but not lost, as the PnP architecture and implementation has progressed and the SDM middleware has been developed.

Four software products were produced in support of this demonstration: mission manager, resource manager, network manager, and the GNC application software. Each of the functional capabilities of these four software modules has been replicated in the current SDM, however, the partitioning and the details of execution are different. The Mission Manager (MM) was the component of the system that handles the mission objectives, requirements, and success criteria. It is in this software that decisions are made regarding the mission phase and the algorithms that must execute at any given time during a mission. This component of the initial demonstration has been replaced with the AFRL activity agents. The Resource Manager (RM) administrates the resource discovery process and maintains the information regarding data descriptions and the system configuration, as well as subsystem health and status. A component of the RM is the local RM that provides an API to resident software applications by abstracting (partitioning for easy reconfiguration) the physical activity needed to access the data while providing a mechanism for error handling and reporting. The original resource manager has been replaced with the processor manager and task manager. The Network Manager (NM) supervises network addressing, routing, protocol, and the interfaces associated with the medium over which data are being supplied. This is the component of the software that will be changed as different mediums and protocols are introduced. This element of the original software has been replaced with the xTEDS and the SPA network protocols.

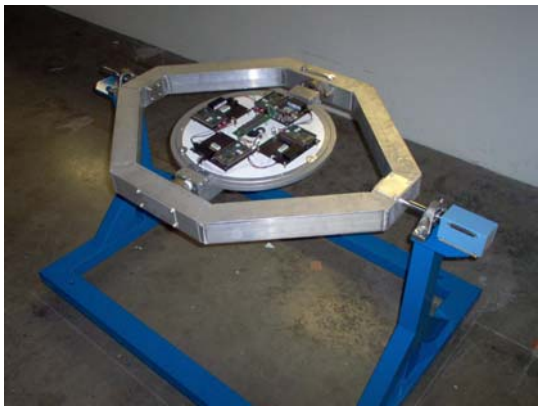

Fig. 4. Demonstration fixture housing self-configuring network of GNC devices

As described above, the Microcosm team developed a software-based product that provided network dependent and independent components. By leveraging commercial off-the-shelf (COTS) networks to the greatest extent possible, a self-configuring, avionics network was created allowing Microcosm to place the emphasis of this work on the GNC system. The COTS networks created a system that includes non-homogeneous types, with bandwidth and protocol variations, and accommodates a multi-mastered system. The data were both pushed by the producers and pulled by the consumers, which yields the most robust implementation possible. Data flows from point-to-point among the various nodes or can be accessed through the resource and data managers. In any implementation, the network supports deterministic timing with real-time data rates of up to 1 megabit/second on the lower level sensor network and 12 megabits/second on the higher level mission network. The self-configuring, avionics network, in conjunction with GNC algorithms that operate on atomic level data (i.e., data at the lowest possible level of functional complexity, such as spacecraft rotation angles), form the basic system for this initial prototypical PnP system.

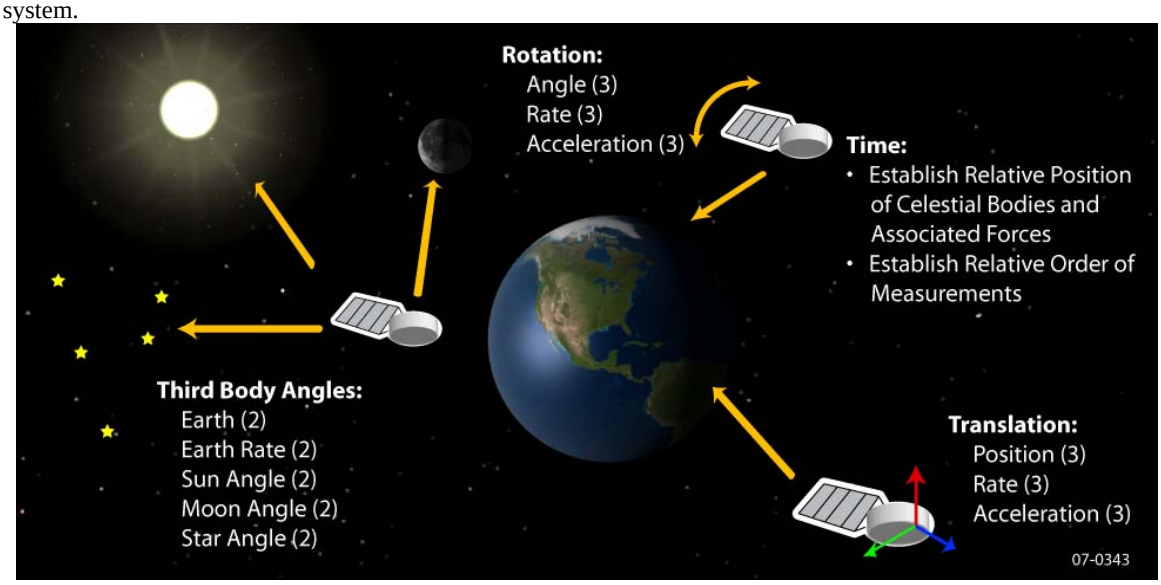

Fig. 5. "Atomic-level" data elements are based on physics rather than specific devices / equipment within a platform

The GNC core algorithms, both for the initial demonstration and for the latest incarnation, were developed to capitalize on the concept of a data centric architecture, where the data is reduced the lowest or "atomic" level. The overall goal is to remove traditional subsystem component boundaries, while defining interfaces in terms that remove vehicle or subsystem component information (i.e. serial number, mass properties) from the implementation. This implementation approach, while antithetical to the traditional approach, will include significant innovation, by creating a processing architecture that is data centric. A data centric architecture implies that the specific devices and/or subsystems are no longer the system drivers, but rather the new driver is selecting the correct vehicle configuration to meet the specific mission needs. If all inputs and outputs are abstracted from the actual components, all that remains is the physics, geometry and tolerances of the measurements and/or actions, labeled as

"atomic data elements" [13][14]. Fig. 5 shows how measured or sensed atomic data has a direct tie to physics and geometry, rather than subsystems or components (e.g., spacecraft motion, rates, expressed in body coordinates).

For the initial demonstration (February 2005), GNC algorithms were hosted on a PC implementing a bang-bang controller. When the controller believed that thruster firings were necessary, it attempted to fire a thruster, represented by LEDs on a separate 8051 processor board. Since on a static mount on the ground a change in attitude does not occur, the controller would continue to request thruster firings. Thus, the system was not closed-loop but demonstrated the ability to achieve a viable solution with varying sensor inputs, which in turn demonstrated the resource discovery, management, and reconfiguration process at a low level. Fault tolerance in the system was achieved both by redundancy of devices and by overdetermined data, which works with the GNC application software because the data expected by the application software was at the atomic data level.

The Microcosm team working with AFRL has continued to refine this PnP GNC architecture and flight software. The control algorithms remain abstracted from the data sets and data is resolved to the atomic level. Additionally, the command and control aspect of the GNC system has been reduced to the atomic level, allowing a user to build-up any capability that is required. Traditional modes such as sun point, ground track, and ground stare, are all achieved using atomic level mode commands. These modes include the following:

- SLEW – Slew to a vector (time, position, velocity)
- TRACK – Maintain pointing to a vector (time, position, velocity)
- RATE – set body rates to a vector (de-tumble)
- STANDBY – transitional or cruise (minimum power consumption)
- MOMENTUM_DUMP – reduce stored momentum

By using this "atomic level" modeling, there may be varying modes of operation that will be performed with respect to varying objects. As shown in Fig. 6, the objects or pointing directions are defined to include planetary objects (Sun, Moon, etc.), Nadir defined as perpendicular to the earth surface, zenith defined as the opposite of nadir. Additionally, the ability to point to a vector (position) or target (state vector including rates) can be accommodated. This leads to an additional concept for the reuse of the PnP GNC FSW. Since the inputs, processing, and outputs are the same for any of the traditional modes, there can also be a single control law. While this may not allow for optimal gains in each of the modes, the overall FSW is quick and easy to assemble, meeting the needs of ORS.

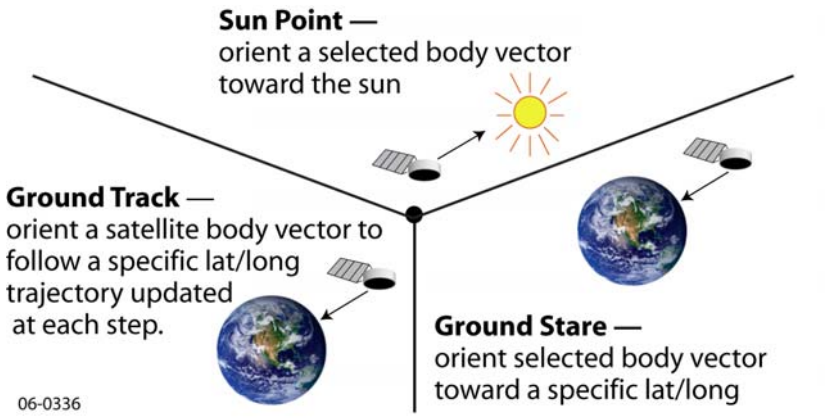

Fig. 6. Atomic Level Moding Facilitates One Control Law Designed to Reduce the Error Between Sensed and Desired State

Since the PnP flight software architecture is predicated on the abstraction of the core algorithm components from the specific sensor and actuator suites, the concept of atomic data elements is augmented with the addition of helper apps. Helper apps provide single functions as services to the core algorithms. At the center of the concept, helper apps make use of abstracted vehicle specific information (system configurations, sensor and actuator mounting locations and the associated coordinate transformations) along with mass properties (e.g., center of mass and pressure) to integrate data from newly discovered sensors and actuators into the ADCS algorithms. The initial type

of helper apps provide a means for translating specific sensor and actuator data into the atomic data elements, referenced to the spacecraft coordinate frame, that are needed by the generic, reusable, core algorithms.

An additional element in the GNC PnP FSW is a planning function. In a traditional development cycle, "what-if" scenarios are examined through endless simulations, typically on the ground, to assure that the space asset can perform the requested mission within the resources of the vehicle. With a PnP vehicle, assembled for responsive missions, these "what-if" scenarios and resource optimization functions must be implemented on board the vehicle for autonomous execution. Thus, a responsive space vehicle must include not only a smart structure, with the associated network infrastructure that supports PnP components and the data centric PnP FSW architecture, it must also provide on-board, autonomous planning and optimization of suggested missions to assure vehicle safety, while accommodating the warfighter requests.

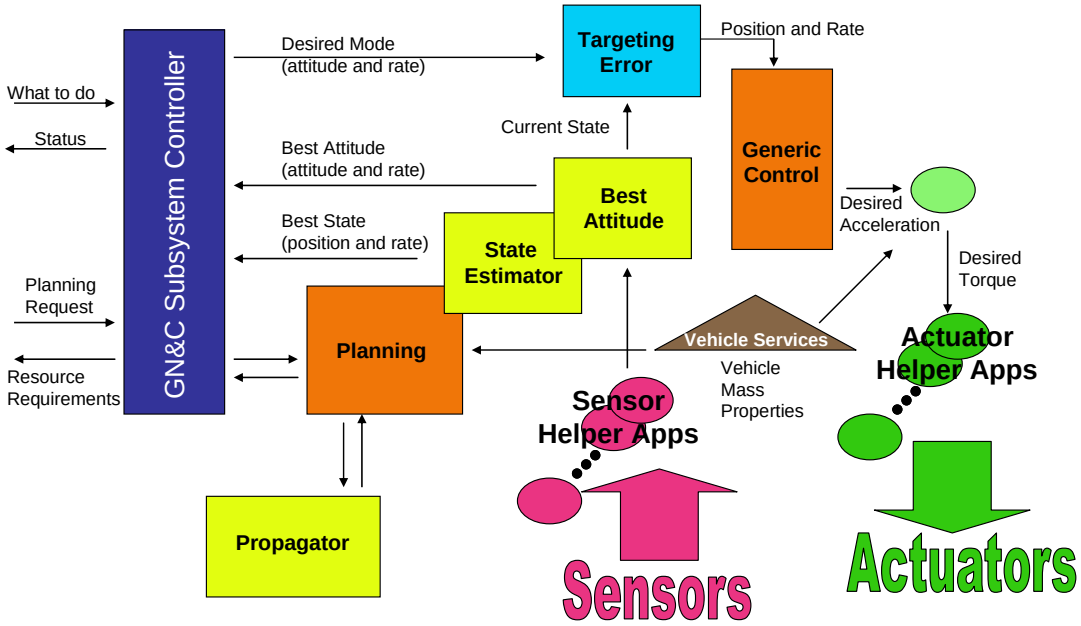

Fig. 7. PnP GNC flight software (FSW) architecture overview

For demonstration at AFRL within the Responsive Space Testbed (RST), the GNC FSW is partitioned as illustrated in Fig. 7. The GNC Subsystem Controller provides a single interface to the rest of the system. There is a planning function and generic control laws each of which provides the traditional functions in a non-traditional implementation. The propagator, state estimator, and best attitude components process all available data within the system and provide the best solution as an output. The targeting error component determines the difference between the vehicle actual state and the desired state. The sensor and actuator helper apps translate data from the supplier format to the PnP required atomic data level elements. The vehicle services, external to the GNC FSW, provides the specific vehicle configuration and mass properties, and the sensor and actuator helper apps, are needed to facilitate core algorithms that have been completely abstract from the specific sensor/actuator suite.

The planning function, as currently implemented provides a basic mechanism to ask "what if" type of questions. The construct of a planning request may include information such as:

- **Mode / Label**
  - Slew, Track, Rate, Standby, Momentum_Dump
- **Object**
  - Sun, Moon, Nadir, Vector, Zenith, Target
- **Boresight (body)**
  - {x,y,z} Track, {rates} Rate
- **Vector (body)**
  - {x,y,z}

- **Offset**
  - {r,p,y}
- **Target (state vector)**
  - {Coordinate System, Time, Position, Velocity, Acceleration}
- **Constraint**
  - Time {seconds}
  - Accuracy {degrees}
  - Power {minimize or watts available}

From this type of information, nearly any GNC activity can be constructed. The GNC function responds with a flag that identifies if the request can be met, then delineates the resources, such as power, bandwidth, and time that are required to meet the requested objective. In the AFRL PnP paradigm, the activity agent that requested the planning information is responsible for determining if the activity should be executed and then issues an operating command (rather than a planning command) to the GNC subsystem controller. For example, the activity agent can determine the minimum amount of time required to achieve a state (time constraint) and will receive back the cost of getting there quickly, typically in terms of power. However, if power is to be conserved, but the activity agent wants to determine if pointing the solar arrays toward the sun for power maintenance can be achieved during the next orbit, the power constraint would be used and the resource requirements would focus on time to achieve the desired state. Additionally, if the activity agent is concerned with how well the spacecraft can point to a specified location on the ground, the constraint would be accuracy, and the resource requirements of interest would include both power and time.

At the RST at AFRL, much of the GNC FSW has been installed and executed. Fig. 8 shows the prototype PnP vehicle that includes smart structures with embedded network and connections, as well as power distribution. AFRL is also developing a PnP Satellite (PnPSat) that is currently being validated with a very limited set of FSW. Based on the spiral development methodology, this FSW will continually be enhanced until full functionality is integrated and validated. The concept of spiral development is important here because not only does it allow for incremental integration, it also show cases many of the PnP concepts that enable rapid assembly and test. Additionally, using the spiral development philosophy, the Microcosm team is proposing other enhancements and upgrades to the PnP GNC FSW.

Fig. 8. “Concept bus”, conceptual progenitor of the Plug-and-play Satellite

The Microcosm team, in conjunction with AFRL and the ISET, are working to continue to develop these enabling technologies that will make ORS a reality. The creation of the RST at AFRL and it’s continued use and enhancement demonstrates the commitment that AFRL is making to PnP and the associated philosophies of modularity and reuse. To move beyond a disruptive technology to a mainstream approach to development, continued focus and effort must be given to this shift in paradigm. The ORS office, AFRL, ISET, and others are continuing the work through funded activities within the areas launch vehicle and ground operations, payload and

rapid payload assembly, and modular spacecraft bus development. Specifically, the Microcosm team has proposed additional work that would enhance the current PnP GNC FSW, without impacting the current performance. These include:

- Establish and implement adaptive state estimation and control laws to auto-tune as the FSW is run
- Creating a more robust autonomous planning function with embedded high fidelity propagator
- Developing a feed-forward control law to compensate for solar array drive torques, payload gimbal torques or to compensate estimated environmental disturbances
- Incorporating observation and control/damping of spacecraft flexible modes
- Integrating multi-use components: using solar array drag torques to support torque balancing (differential array angles for center-of-pressure adjustment) or momentum unloading, or fluid transfers to trim center-of-gravity

CONCLUSION

To meet the evolving needs of Operationally Responsive Space the Microcosm team, under AFRL support and direction, has developed a plug-and-play GNC flight software (FSW) architecture and prototype implementation. This software is being evaluated through flight software-in-the-loop and hardware-in-the-loop testing at AFRL's responsive space testbed (RST). A minimal set of the GNC FSW is scheduled to fly on the PnP Satellite (PnPSat) which is in final testing at the time of this writing. GNC applications are typically hard-real-time, and thus, the prototype FSW is being evaluated for real world viability for its ability to approximate adequate ("good enough") determinism. It is through continued experimentation and demonstration that practicality of new philosophies and approaches to spacecraft development will be refined and ultimately accepted.

REFERENCES

[1] Lyke, J., Cannon, S., Fronterhouse, D., Lanza, D., and Byers, T. "A Plug-and-play System for Spacecraft Components Based on the USB Standard", *Proceedings of the 19th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 8-11 August, 2005.

[2] Lyke, J.; Space-Plug-and-Play Avionics (SPA): A Three-Year Progress Report, Proceedings *of the AIAA Infotech Conference*, 7-9 May 2007, Rohnert Park, CA.

[3] Fronterhouse, Don; Lyke, James; Achramowicz, Steve.; "Plug-and-play Satellite", *Proceedings of the AIAA Infotech Conference*, 7-9 May 2007, Rohnert Park, CA.

[4] Christensen, Clayton M. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Boston, Mass.: Harvard Business School Press, 1997.

[5] Marco Cáceres , "Cost overruns plague military satellite programs", *Aerospace America* (publication of AIAA), January 2006, pp 18-20, 23.
(http://www.aiaa.org/aerospace/images/articleimages/pdf/AA_Jan06_II.pdf , accessed 10 January 2008).

[6] "DoD Stands up Joint Space Office", *Air Force Link*, press release, 22 May 2007
(http://www.af.mil/news/story.asp?id=123054214, accessed 20 May 2008).

[7] "Plan for Operationally Responsive Space", A Report to Congressional Defense Committees, 17 April 2007, available through http://www.acq.osd.mil/nsso/ors/ors.htm (accessed 20 May 2008).

[8] "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communications Protocols and Transducer Electronic Data Sheet (TEDS) Formats", Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ 08855, 1997.

[9] McGuirk, P., Rakow, G., Kemmerly, K., Jaffe, P., Klar, R., and Betrand, A. "Spacewire Plug-and-Play", *Proceedings of the AIAA Infotech Conference*, 7-9 May 2007, Rohnert Park, CA.

[10] Cannon S, "Responsive Space Plug & Play with the Satellite Data Model", *Proceedings of the AIAA Infotech 2007 Conference*, Rohnert Part, CA, May 2007.

[11] Graven, Paul, Yegor Plam, L. Jane Hansen, Seth Harvey, "Implementing Plug-and-Play Concepts in the Development of an Attitude Determination and Control System to Support Operationally Responsive Space," presented at the IEEE Aerospace Conference, Big Sky, Montana, March 1-8, 2008.

[12] Graven, Paul, L. Jane Hansen, "A Guidance, Navigation and Control (GNC) Implementation of Plug-and-Play for Responsive Spacecraft," presented at the AIAA Infotech@Aerospace 2007 Conference and Exhibit, Rohnert Park, CA, 7 – 10 May 2007

[13] Hansen, L. Jane, Jon Pollack, Paul Graven, Yegor Plam, "Plug-and-Play for Creating 'Instant" GNC Solutions," presented at the21st annual AIAA/USU Conference on Small Satellites, Logan Utah, Aug. 13-16, 2007.

[14] Graven, Paul, Richard Van Allen, L. Jane Hansen, Jon Pollack, "Achieving Responsive Space: The Viability of Plug-and-Play in Spacecraft Development," presented at the 29th annual AAS Guidance and Control Conference, Breckenridge, CO, Feb. 4-8, 2006.